CREATE OR REPLACE FUNCTION std5\_33.func\_load\_traffic(p\_table text, p\_partition\_key text, p\_start\_date timestamp, p\_pxf\_table text, p\_user\_id text, p\_pass text)

RETURNS int4

LANGUAGE plpgsql

VOLATILE

AS $$

declare

v\_ext\_table text;

v\_temp\_table text;

v\_table\_oid int4;

v\_sql text;

v\_pxf text;

v\_result int;

v\_dist\_key text;

v\_params text;

v\_where text;

v\_load\_interval interval;

v\_start\_date date;

v\_start\_date\_tg text;

v\_start\_date\_src text;

v\_end\_date date;

lv\_res int4;

v\_cnt int8;

lt\_tab record;

v\_check text;

begin

v\_ext\_table = p\_table||'\_ext';

v\_temp\_table = p\_table||'\_tmp';

select c.oid

into v\_table\_oid

from pg\_class as c inner join pg\_namespace as n on c.relnamespace = n.oid

where n.nspname||'.'||c.relname = p\_table

limit 1;

if v\_table\_oid = 0 or v\_table\_oid is null then

v\_dist\_key = 'DISTRIBUTED RANDOMLY';

else

v\_dist\_key = pg\_get\_table\_distributedby(v\_table\_oid);

end if;

select coalesce(' with (' ||array\_to\_string(reloptions, ', ')|| ')','')

from pg\_class

where oid = p\_table::regclass

into v\_params;

v\_load\_interval = '1 month'::interval;

EXECUTE 'DROP EXTERNAL TABLE IF EXISTS '||v\_ext\_table;

v\_pxf = 'pxf://'||p\_pxf\_table||'?PROFILE=Jdbc&JDBC\_DRIVER=org.postgresql.Driver&DB\_URL=jdbc:postgresql://192.168.214.212:5432/postgres&USER='||p\_user\_id||'&PASS='||p\_pass;

RAISE NOTICE 'PXF CONNECTION STRING: %', v\_pxf;

v\_sql = 'CREATE EXTERNAL TABLE '||v\_ext\_table||'(plant bpchar,

date bpchar,

time bpchar,

frame\_id bpchar,

quantity int4)

LOCATION ('''||v\_pxf||'''

) ON ALL

FORMAT ''CUSTOM'' (FORMATTER=''pxfwritable\_import'')

ENCODING ''UTF8''';

EXECUTE v\_sql;

RAISE NOTICE 'EXTERNAL TABLE IS: %', v\_sql;

-- ласт дата в текущей traffic - целевая NULL

v\_sql = 'SELECT max('||p\_partition\_key||'::date) FROM '||p\_table;

execute v\_sql into v\_start\_date\_tg;

-- ласт дата в traffic\_ext == gp.traffic - источник 2021-02-28

v\_sql = 'SELECT max(to\_date('||p\_partition\_key||', '||quote\_literal('DD.MM.YYYY')||')) as '||p\_partition\_key||' FROM '||v\_ext\_table;

execute v\_sql into v\_start\_date\_src;

-- если пустая traffic,то старт дата из параметра '2021-01-01', енд дата - ласт дата в источнике(traffic\_ext == gp.traffic)

-- если что-то есть в traffic то старт дата - это ласт дата в текущей traffic (v\_start\_date\_tg), а енд дата - текущая дата

if v\_start\_date\_tg = '' or v\_start\_date\_tg is NULL

then -- если пустая

raise NOTICE '->>>>>>>>>>>>>>>>>>>>> IF BLOCK';

v\_start\_date := DATE\_TRUNC('month', cast(p\_start\_date as date)); -- будет 2021-01-01

v\_end\_date = v\_start\_date\_src; -- будет 2021-02-28

else

v\_start\_date := cast(v\_start\_date\_tg as date)+interval '1 day';

v\_end\_date = current\_date;

end if;

while v\_start\_date <= v\_end\_date -- 2021-01-01 <= 2021-02-28

loop

v\_sql := 'DROP TABLE IF EXISTS '|| v\_temp\_table ||';

CREATE TABLE '|| v\_temp\_table ||' (LIKE '||p\_table||') ' ||v\_params||' '||v\_dist\_key||';';

raise notice 'TEMP TABLE IS: %', v\_sql;

execute v\_sql;

-- date >= 2021-01-01 and date < 2021-01-01+ 1 month = 2021-02-01

v\_where = 'to\_date('||p\_partition\_key||', '||quote\_literal('DD.MM.YYYY')||') >='''||v\_start\_date||'''::date AND to\_date('||p\_partition\_key||', '||quote\_literal('DD.MM.YYYY')||') < '''||v\_start\_date||'''::date+interval '||quote\_literal('1 month')||'';

raise NOTICE 'v\_where: %', v\_where;

v\_sql = 'INSERT INTO '|| v\_temp\_table ||' SELECT plant,

to\_date(date, '||quote\_literal('DD.MM.YYYY')||') as date,

TO\_TIMESTAMP(time, '||quote\_literal('HH24MISS')||') ::TIME as time,

frame\_id,

quantity

FROM '||v\_ext\_table||' WHERE '||v\_where;

raise notice 'INSERT IS: %', v\_sql;

execute v\_sql;

-- проверка ,если есть хотя бы что-то в traffic\_tmp то в v\_check

v\_sql = 'SELECT \* FROM '|| v\_temp\_table ||' LIMIT 1';

execute v\_sql into v\_check;

-- меняем партицию

if v\_check is not null

then

-- traffic 2021-01-01

v\_sql = '

ALTER TABLE '||p\_table||'

EXCHANGE PARTITION FOR (DATE '''||v\_start\_date||''')

WITH TABLE '|| v\_temp\_table ||'

WITH VALIDATION;';

raise notice 'EXCHANGE PARTITION SCRIPT: %', v\_sql;

execute v\_sql;

else

v\_start\_date = v\_end\_date;

end if;

v\_sql := 'DROP TABLE IF EXISTS '|| v\_temp\_table;

execute v\_sql;

v\_start\_date = v\_start\_date + v\_load\_interval;

end loop;

v\_result = 1;

return v\_result;

end;

$$

EXECUTE ON ANY;

Что делает эта функция по шагам и как её можно упростить, преобразовав поступающие на вход данные столбца date к типу данных date внутри функции?

Эта функция выполняет загрузку данных из внешней таблицы `p\_pxf\_table` во временную таблицу `v\_temp\_table`, затем обменивает партиции между таблицами `p\_table` и `v\_temp\_table` на основе даты, указанной в `p\_partition\_key`. Шаги функции: 1. Создание внешней таблицы `v\_ext\_table` на основе данных из `p\_pxf\_table`. 2. Получение последней даты из текущей таблицы `p\_table` и из внешней таблицы `v\_ext\_table`. 3. Определение стартовой и конечной дат загрузки в зависимости от наличия данных в текущей таблице `p\_table`. 4. Цикл загрузки данных из внешней таблицы во временную таблицу по месяцам. 5. Проверка наличия данных во временной таблице. 6. Обмен партиций между текущей таблицей и временной таблицей. 7. Удаление временной таблицы и переход к следующему месяцу. Чтобы упростить функцию и избежать проблем с преобразованием типов данных, можно изменить тип данных столбца `date` во внешней таблице на `DATE`, а не на `bpchar`. Также, можно использовать стандартные функции преобразования типов данных внутри SQL запросов, чтобы избежать лишних преобразований и упростить логику функции.